

**REMARKS**

Claims 1 and 23-42 are pending. No claims stand allowed. Claims 1 and 23-42 were rejected in an office action dated February 26, 2003. Claims 1, 32, 34, 35, 36, 38, 40, 41, and 42 have been amended to further particularly point out and distinctly claim subject matter regarded as the invention. Support for the amendment to claims 1, 32, 34, 35, 36, 38, 40, 41, and 42 is provided in the original specification, page 8 lines 23-26, page 11 lines 16-20, page 16 line 10 – page 20 line 23, page 25 lines 3-17, page 27 line 1 – page 28 line 12, and FIGS. 5, 6, 7, 8, 13, 16, 17, and 18. No “new matter” has been added by the Amendment.

**The 35 U.S.C. § 102 Rejection**

Claim 32 stands rejected under 35 U.S.C. § 102(b) as being allegedly anticipated by “Java Card 2.0 Programming Concepts” by Sun Microsystems, Inc.<sup>1</sup> This rejection is respectfully traversed. As independent claim 32 has been amended to include limitations not shown or suggested by the cited art, the Examiner’s objections should now be rendered moot.

According to the M.P.E.P., a claim is anticipated under 35 U.S.C. § 102(a), (b) and (e) only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference.<sup>2</sup> “The identical invention must be

---

<sup>1</sup> Office Action dated February 26, 2003, ¶ 3.

<sup>2</sup> Manual of Patent Examining Procedure (MPEP) § 2131. See also *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987).

shown in as complete detail as contained in the claim.”<sup>3</sup> “The elements must be arranged as required by the claim...”<sup>4</sup>

The Applicants respectfully submit that each and every element as set forth in amended independent claim 32 is not found in the cited reference.

Claim 32 as amended recites:

A method of operating a small footprint device that includes a processing machine, wherein program modules are executed on the processing machine, the method comprising:  
separating contexts using a context barrier, said context barrier configured to isolate said contexts and to control the object-oriented access of a program module executing in another context, said separating further comprising:  
preventing said access if said access is unauthorized; and  
enabling said access if said access is authorized;  
executing groups of one or more program modules in separate contexts, objects of a program module associated with a particular context; and  
permitting access to information across said context barrier by bypassing said context barrier using a global data structure.

Amended independent claim 32 specifies object-by-object security not disclosed or suggested by the cited art. Specifically, claim 32 specifies executing groups of one or more program modules in separate contexts, *objects of a program module associated with a particular context*. Amended independent claim 32 also specifies a context barrier for separating and isolating said contexts, said context barrier configured to isolate said contexts and to control the *object-oriented* access of a program module executing in one context to information and/or a program module executing in another context.

---

<sup>3</sup>*Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989).

<sup>4</sup> *Id.*

Whereas the cited reference discloses “an applet firewall prevents one *applet* from accessing the contents or behavior of objects *owned by other applets*.”<sup>5</sup> The cited reference also states:

Every object (class instance or array) on the card is owned by the applet which instantiated it, that is, the applet which was active at the time the object was created. The owning applet always has full privileges to use and modify the object.

The applet firewall ensures that no other *applet* may use, access, or modify the contents of an object *owned by another applet* except as described in this section. This does not restrict another *applet* from having a reference to such an object, but that *applet* cannot invoke methods on the object or get or set its field contents.<sup>6</sup>

Since all elements of amended independent claim 32 are not disclosed in the cited references, the § 102 reference is unsupported by the art and should be withdrawn.

#### The 35 U.S.C. § 103 Rejection

Claims 1, 23-31, and 33-42 stand rejected under 35 U.S.C. §103(a) as being allegedly unpatentable over Java Card 2.0 Programming Concepts” by Sun Microsystems, Inc.<sup>7</sup> This rejection is respectfully traversed. As independent claims 1, 32, 34, 35, 36, 38, 40, 41, and 42 have been amended to include limitations not shown or suggested by the cited art, the Examiner’s objections should now be rendered moot.

According to the Manual of Patent Examining Procedure (M.P.E.P.),

---

<sup>5</sup> Java Card 20. Programming Concepts, October 15, 1977, Sun Microsystems, Inc., § 2.7. (emphasis added)

<sup>6</sup> *Id.* (emphasis added)

<sup>7</sup> Office Action ¶ 5.

To establish a *prima facie* case of obviousness, three basic criteria must be met. First there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, not in the applicant's disclosure.<sup>8</sup>

The Applicants respectfully submit that the cited art does not teach or suggest all claim limitations.

The sandbox model described by the cited art is fundamentally different from the present invention as disclosed and claimed. Briefly, protection domains in the sandbox model are strictly code-based whereas the contexts disclosed in the present invention are object-based. So the protection domains of the sandbox model are not equivalent to the contexts in the present invention. As stated in the Application, "the present invention addresses the need to allow object-oriented interaction between selected execution contexts only in safe ways via fast efficient peer to peer communications..."<sup>9</sup> The cited art cannot contribute to this need because it is code-based and not object-oriented.

The following definitions may be helpful in understanding the difference between the present application and the sandbox model described in the cited art: A class is the prototype for an object in an object-oriented language. A class may also be considered a set of objects that share a common structure and behavior. The structure of a class is determined by the class variables that represent the state of an object of that class and the

---

<sup>8</sup> M.P.E.P § 2143.

<sup>9</sup> Specification at p. 8 lines 23-26.

behavior is given by a set of methods associated with the class. Objects are unique instances (a kind of copy) of a data structure defined according to the template provided by its class. Each object has its own values for the variables belonging to its class and can respond to the messages (methods) defined by its class. Multiple objects may share the same class.

The cited reference states:

Every object (class instance or array) on the card is owned by the applet which instantiated it, that is, the applet which was active at the time the object was created. The owning applet always has full privileges to use and modify the object.

The applet firewall ensures that no other *applet* may use, access, or modify the contents of an object *owned by another applet* except as described in this section. This does not restrict another *applet* from having a reference to such an object, but that *applet* cannot invoke methods on the object or get or set its field contents.<sup>10</sup>

This is in contrast to the Application, which states:

Every object is associated with one particular context. That context is said to own each object that is associated with it. The runtime system 740 provides a means for uniquely identifying contexts, and a means for specifying and identifying the currently executing context. The object system 750 provides a mechanism for associating objects with their owning *contexts*.<sup>11</sup>

Pages 17-20 of the application define the relationship between objects, data, entities and execution contexts. Specifically, the application teaches:

As with data, a security check enforced by the operating system may use the identity of the principal, the identity of the entity, and/or the type of action. Furthermore, the entity, being active, can perform its own additional security checks. These can be as arbitrarily complex as one desires, and can make use of the identity of the Principal, the identity of the entity itself, the action, and/or any

---

<sup>10</sup> *Id.* (emphasis added)

<sup>11</sup> Specification at p. 11 lines 16-20. (emphasis added)

other information that is available.<sup>12</sup>

Note that "owning applet" is not listed.

The appendix of the Application describes an embodiment of the concept of contexts as follows:

Firewalls essentially partition the Java Card platform's object system into separate protected *object* spaces called contexts.<sup>13</sup>

The sandbox model described the cited reference is code-based and there is a mapping from the applet code to the protection domains and their permissions, whereas the present application is object-based, so there is no such mapping but a completely different concept. According to the Application, security checks may use the identity of the principal, the identity of the entity, and/or the type of action, but no mention is made of basing the security check on the applet code. Indeed, it would seem using a code-based approach is inconsistent with an object-based approach because in an object-based approach, multiple objects may share or belong to the same class and yet be in different contexts. The code-based approach of the sandbox model would require objects sharing the same class to have the same permissions because the objects share a common structure and behavior (the code).

This is in contrast to embodiments of the present invention as claimed, where a context barrier isolates and separates data and applications of different contexts from each other but allows controlled *object-oriented* access of a program module executing in

---

<sup>12</sup> Specification at p. 17 lines 16-24.

one context to information and/or a program module executing in another context, where *objects of a program module are associated with a particular context*, and where a global data structure permits one program module to access information from another program module by bypassing the context barrier. With this Amendment, independent claims 1, 32, 34, 35, 36, 38, 40, 41, and 42 have been amended to make this distinction more clear.

As independent claims claims 1, 32, 34, 35, 36, 38, 40, 41, and 42 have been amended to include limitations not shown or suggested by the cited art, the 35 U.S.C. § 103 rejection as applied to claims claims 1, 32, 34, 35, 36, 38, 40, 41, and 42 is unsupported by the art and should be withdrawn.

#### Dependent Claims

Claims 23-31 depend from claim 1. Claims 33, 35, 37, and 39 depend from claims 32, 34, 36, and 38, respectively. The base claims being allowable, the dependent claims must also be allowable.

In view of the foregoing, it is respectfully asserted that the claims are now in condition for allowance.

#### Request for Allowance

---

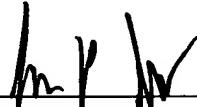
<sup>13</sup> JCRE 2.1 Spec, Sect. 6.1.1. (emphasis added)

It is believed that this Response places the above-identified patent application into condition for allowance. Early favorable consideration of this application is earnestly solicited.

If, in the opinion of the Examiner, an interview would expedite the prosecution of this application, the Examiner is invited to call the undersigned attorney at the number indicated below.

Respectfully submitted,  
THELEN REID & PRIEST, LLP

Dated: June 5, 2003

  
\_\_\_\_\_  
John P. Schaub  
Reg. No. 42,125

THELEN REID & PRIEST LLP  
P. O. Box 640640  
San Jose, CA 95164-0640  
Tel: (408) 292-5800